PROGETTO 2022/2023

IL PROGETTO SERVE PER

- Farvi sperimentare le tecniche viste a lezione su scala più grande di quanto possibile in una lezione di 3 ore, in direzione di quello che succede nel mondo reale
- I progetti reali sono spesso MOLTO più grandi
- Mettervi alla prova in uno scenario che lascia spazio alla vostra fantasia, ma pone anche alcuni vincoli

VALUTAZIONE DEL PROGETTO

- Il progetto è parte integrante dell'esame ed è **obbligatorio**
- Ha un voto massimo di 8 punti che si sommano al voto dello scritto (max 24)
- Il lavoro viene svolto in gruppi, ma la valutazione del progetto è **individuale**
    - Studenti dello stesso gruppo possono prendere voti diversi!
- Il progetto **si consegna una volta sola** ed il voto è valido per l'intero anno accademico (anche per quello successivo se le regole non cambiano)
- Chi non ottiene almeno 2 punti alla discussione orale dovrà presentare un nuovo progetto

GRUPPI

- Il progetto si svolge in gruppi di 3/4 persone, ma il voto non è di gruppo
- NON è possibile fare gruppi di 5 o più persone
- Sono ammessi **progetti individuali in casi eccezionali** (ma il carico di lavoro non diminuisce)
- Auto-organizzatevi per la creazione di gruppi
- Comunicherete i componenti del gruppo in fase di prenotazione e/o discussione del progetto

QUANDO CONSEGNARE IL PROGETTO?

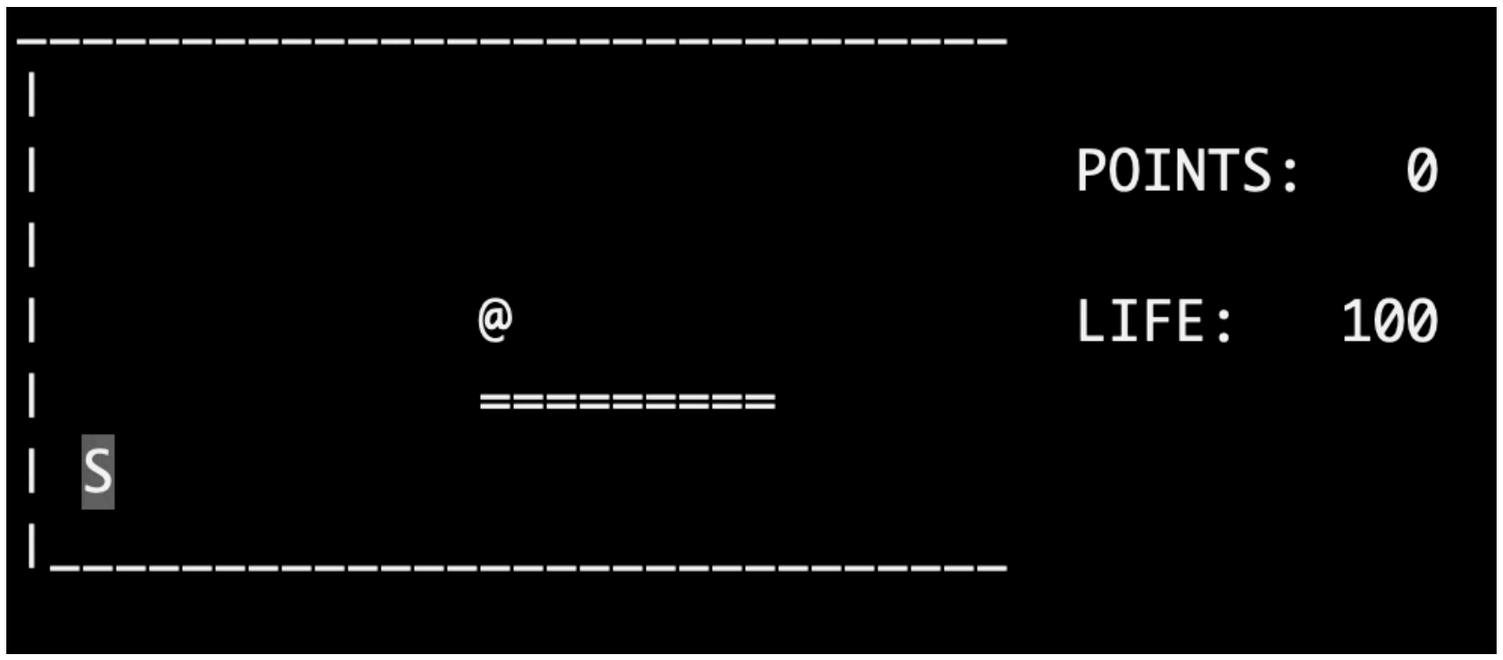
- Il progetto potrà essere presentato entro il 28/02/2024
- Orientativamente ci sarà un appello ogni appello dello scritto (uno/due per sessione)
- Le date degli appelli verranno comunicate tramite Virtuale
- Se ci sono molto richieste, possiamo fare ulteriori date, fuori dalle date prestabilite

CONSEGNA E DISCUSSIONE

- Il progetto si consegna **una settimana** prima della discussione
- TUTTI i membri del gruppo devono partecipare alla discussione
- Il gruppo consegna:
  - Codice (sorgenti e binari)
  - File README
  - Screen recording che mostri l'esecuzione del gioco
  - Breve relazione (3/4 pagine) in cui si descrivono le principali scelte nell'implementazione del progetto
- Consegna tramite Virtuale

CONTENUTI DEL GIOCO

- Si richiede di implementare un gioco platform in grafica ASCII
- Sviluppato su più livelli
- A punteggio
- Il protagonista viene controllato da tastiera



REQUISITI MINIMI E OBBLIGATORI / SVOLGIMENTO DEL GIOCO

1. La mappa deve avere una grafica ASCII, **è possibile utilizzare solo le librerie curses/ncurses.h**
2. Lo schema del livello è **scelto random da un insieme di livelli pre-definiti (o generati random)**.
   Questo significa **non si parte mai dallo stesso schema.**
3. Man mano che si procede la difficoltà aumenta (più nemici, meno potenziamenti)
4. Non esistono traguardi
5. Durante la partita, il protagonista può tornare indietro di schema e trovare esattamente tutto come lo ha lasciato
6. Devono essere presenti **diversi tipi di nemici e una valuta**
7. **La valuta** può essere utilizzata per acquistare potenziamenti (vita, poteri, armi)

REQUISITI MINIMI E OBBLIGATORI / SVOLGIMENTO DEL GIOCO

8. Il gioco deve quindi prevedere un **livello market** (e.g., a inizio partita) in cui acquisire vita aggiuntiva, poteri e/o armi.
9. Se il protagonista muore, mantiene i bonus, poteri e/o armi che ha acquistato e alla prossima partita partirà da uno schema random (vedi punto 2) e una difficoltà proporzionata ai potenziamenti che ha acquisito
10. Il giocatore umano controlla il protagonista
11. Nel percorso il protagonista incontra nemici i quali possono **togliere** vita colpendo o entrando a contatto con il protagonista
12. I nemici sono controllati dal computer: si possono muovere e sparare, possono venire uccisi dal protagonista (es. sparando)
13. Game over quando la vita scende a 0

IMPOSTAZIONE DEL PROGETTO

- Il progetto deve essere realizzato usando le classi
- Le stanze e gli elementi in una stanza devono essere gestiti tramite strutture dati dinamiche (prevedendo, quindi, inserimento e rimozione)
- Il progetto è organizzato in più file
- Ad ogni classe corrispondono due file: NomeClasse.cpp e NomeClasse.hpp

ESEMPIO DIVISIONE DEL LAVORO ALL'INTERNO DI UN GRUPPO

- Componente #1: si occupa delle mappe
- Componente #2: implementa personaggi e nemici
- Componente #3: si occupa della interazione con la libreria grafica
- Componente #4: si fa carico dei potenziamenti, del mercato e del salvataggio di stato

ESEMPIO DI FILE

NomeClasse.hpp

```
class NomeClasse{
protected:
    int field;
    ...
public:
    ...
    void method();
    ...
};
```

NomeClasse.cpp

```
#include "NomeClasse.hpp"

void NomeClasse::method(){
    // do something
}
```

IN OGNI FILE IN CUI SI USA IL TIPO “NOMECLASSE” BISOGNA IMPORTARE NOMECLASSE.HPP

LETTURA DA FILE

- Per leggere e scrivere su file bisogna includere fstream
- ifstream è un **input file stream**: legge dati da un file

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
     ifstream inputFile; /* Dichiarazione di tipo */
     inputFile.open("file.txt");
     char ch;
     /* lettura dati */
     while(!inputFile.eof()){
          inputFile.get(ch);
          cout << ch << endl;
     }
     inputFile.close();
     return 0;
}
```

- **while**(!inputFile.eof()) identifica un ciclo che finisce quando file termina
- In questo contesto si tratta di un file testuale, possiamo adattare a qualsiasi tipo di dato
- Tramite la funzione **get** si memorizza il carattere corrente del file nella variabile ch
- Alla fine del ciclo dobbiamo chiudere lo stream, ovvero il flusso di dati, proveniente dal file in ingresso, tramite l'istruzione **inputFile.close()**;

SCRITTURA SU FILE

- Per leggere e scrivere su file bisogna includere fstream
- ofstream è un **output file stream**: scrive dati su un file

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {

    ofstream outputFile; /* Dichiarazione di tipo */
    outputFile.open("file2.txt"); /* Apertura del file */
    outputFile << "Prova di scrittura su file";
    outputFile.close();

    return 0;
}
```

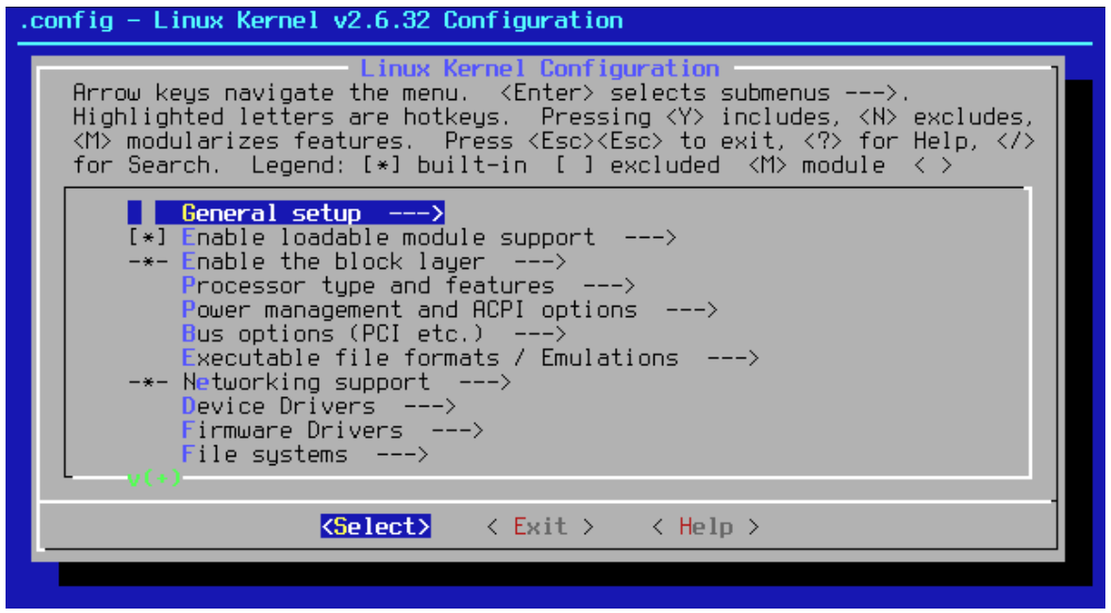
- Se il file da aprire in scrittura non è presente ne verrà creato uno vuoto con il nome specificato
- Possiamo stampare una frase su **outputFile** semplicemente ricalcando la sintassi del comando **cout**
- Per scrivere sul file senza eliminarne il contenuto già presente, è sufficiente aprire il file con il comando:
  outputFile.open(“file2.txt",ios::*app*);

LIBRERIA CURSES/NCURSES

- **ncurses** (new curses) è una libreria utilizzata per consentire al programmatore di scrivere TUI (Text-based User Interfaces).

- È un insieme di strumenti per lo sviluppo di applicazioni "GUI-like" che vengono eseguite tramite terminale.

- ncurses è un'emulazione di software libero delle **curses**.



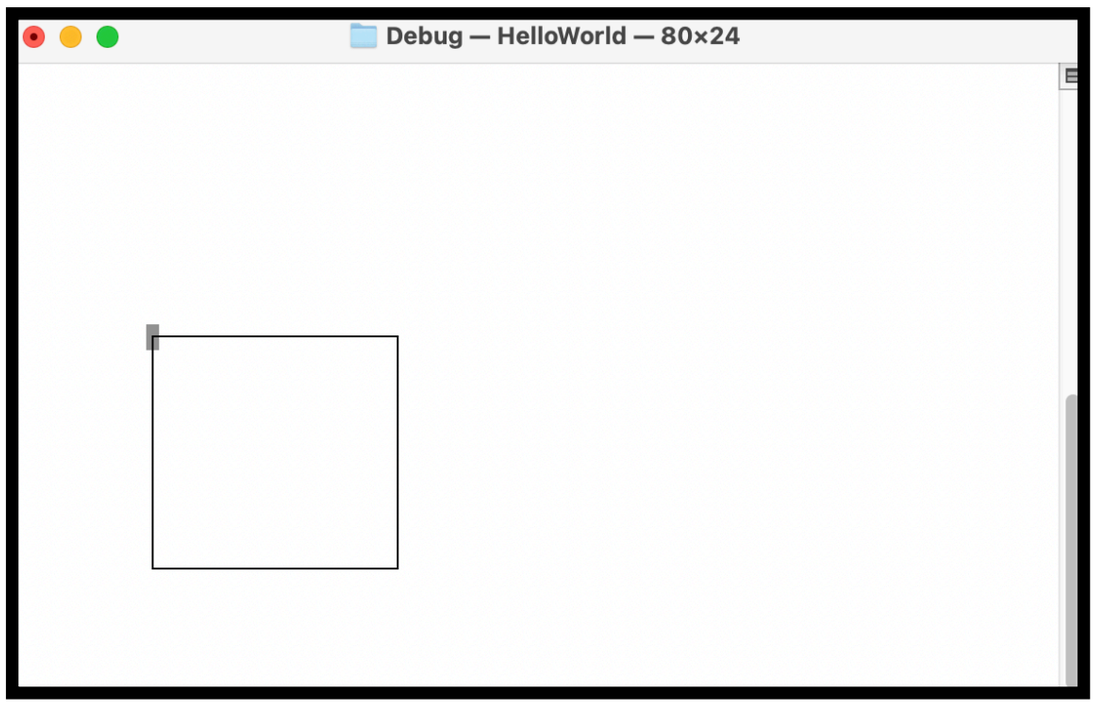
Esempio di applicazione creata con curses

LIBRERIA CURSES/NCURSES

- Esempio dell'utilizzo della libreria

HelloWorld.cpp

```cpp
#include <ncurses.h>
#include <iostream>

using namespace std;

int main(int argc, char ** argv){
    //initializes the screen
    initscr();

    int height, width, start_y, start_x;

    height = 10;
    width = 20;
    start_y = 10;
    start_x = 10;

    WINDOW * win = newwin(height, width, start_y, start_x);

    //refreshes the screen
    refresh();

    //show the window
    box(win, 0, 0);
    wrefresh(win);

    //wait for user input like "cin >> a;"
    getch();

    //deallocates memory and ends ncurses
    endwin();
}
```



- Qui potete trovare una guida per l'utilizzo della libreria: https://www.youtube.com/playlist?list=PL2U2TQ__OrQ8jTf0_noNKtHMuYlyxQl4v

LIBRERIA CURSES/NCURSES

Funzioni principali:

- **initscr()** : inizializza screen
- **endwin()** : dealloca memoria e chiude il programma
- **newwin(height, width, start_y, start_x)** : crea una nuova finestra
- **refresh()** : refresh dello screen con le informazioni aggiuntive
- **move(x, y)** : muove il cursore nella posizione indicata (x, y)
- **clear()** : cancella tutto ciò che è presente sullo screen

LIBRERIA CURSES/NCURSES – LINUX

- Installazione libreria:

sudo apt-get install libncurses5-dev libncursesw5-dev

- Compilazione:

g++ HelloWorld.cpp -lncurses -o HelloWorld

- Esecuzione:

./HelloWorld

LIBRERIA CURSES/NCURSES – MACOS

- Installazione libreria:

brew install ncurses

- Compilazione:

g++ HelloWorld.cpp -lncurses -o HelloWorld

- Esecuzione:

./HelloWorld

LIBRERIA CURSES/NCURSES – WINDOWS

- Da MinGW selezionare i seguenti pacchetti:

libncurses

libpdcurses

ncurses

pdcurses

- Compilazione:

```
g++ -I/mingw64/include/ncurses -o HelloWorld HelloWorld.cpp -lncurses -L/mingw64/bin -static
```

- Esecuzione:

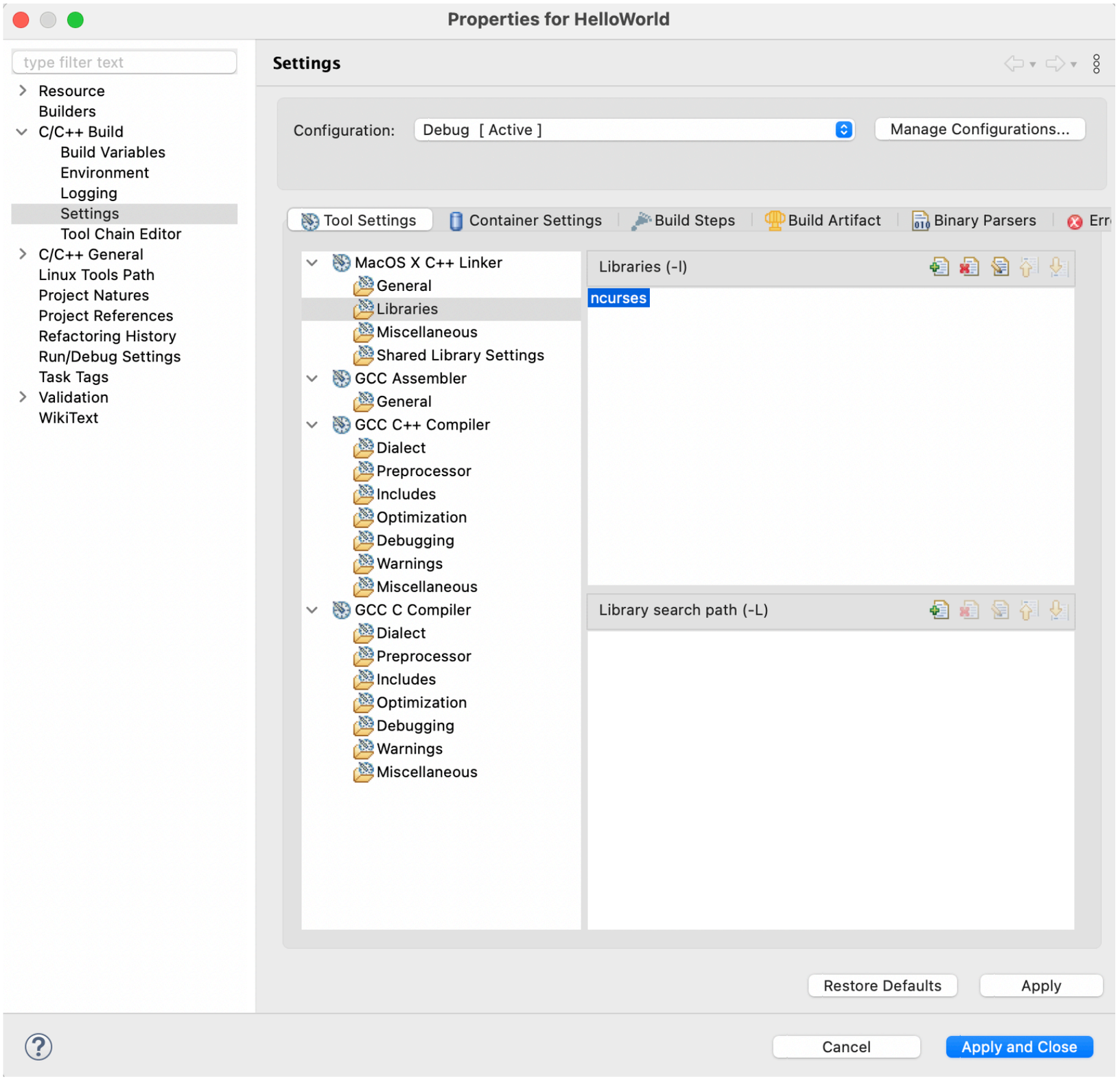
```
HelloWorld
```

oppure

```
./HelloWorld
```

Nel caso non funzionassero i comandi è possibile eseguire il programma facendo doppio click sul file

LIBRERIA CURSES/NCURSES – COMPILAZIONE CON ECLIPSE



- Per compilare i programmi utilizzando Eclipse bisogna aggiungere il link della libreria nelle impostazioni di Build

- Project -> Properties -> C/C++ Build -> Settings -> MacOS X C++ Linker -> Libraries
- Aggiungete ncurses in Libraries (-l)
- Avviate la compilazione

- Eclipse creerà l'eseguibile **all'interno della cartella Debug**
- Eseguite il programma **utilizzando il terminale**